Presented at the International Joint Conference on Neural Networks,
Washington, D. C., June 18-22, 1989
Reproduced on pages I-689-692 of the Proceedings

RAPID TRAINING OF HIGHER-ORDER NEURAL NETWORKS FOR INVARIANT PATTERN RECOGNITION

Max B. Reid, Lilly Spirkovska, and Ellen Ochoa
NASA Ames Research Center
Intelligent Systems Technology Branch
Mail Stop 244-4
Moffett Field, CA 94035

Abstract: We demonstrate a second-order neural network that has learned to distinguish between two objects, regardless of their size or translational position, after being trained on only one view of each object. Using an image size of 16 x 16 pixels, the training took less than 1 minute of run time on a Sun 3 workstation. 100% recognition accuracy was achieved by the resulting network for several test-object pairs, including the standard T-C problem, for any translational position and over a scale factor of 5.

The second-order network takes advantage of known relationships between input pixels to build invariance into the network architecture. The use of a third-order neural network to achieve simultaneous rotation, scale and position invariance is described. Because of the high level of invariance and rapid, efficient training, initial results show higher-order neural networks (HONNs) to be vastly superior to multi-level, first-order networks trained by back-propagation for applications where invariant pattern recognition is required.

Introduction

Pattern recognition requires the nonlinear separation of pattern space into subsets representing the objects to be identified. Early research into neural networks, or perceptrons, concentrated on defining their potential for nonlinear discrimination.[1,2] It was found that a single layer, first-order neural network can only perform linear discrimination. However, either multilayer, first-order networks or single layer networks of higher order can provide the desired nonlinear separation.[2]

The activation level of an output node in a first-order neural network is determined by an equation of the form:

$$y_i = \Theta(\Sigma_j w_{ij} x_j) \qquad (1)$$

where Θ is a nonlinear threshold function, the x_j are the excitation values of the input nodes, and the interconnection matrix elements w_{ij} determine the weight that each input is given in the summation. A simplified diagram is shown in Figure 1.

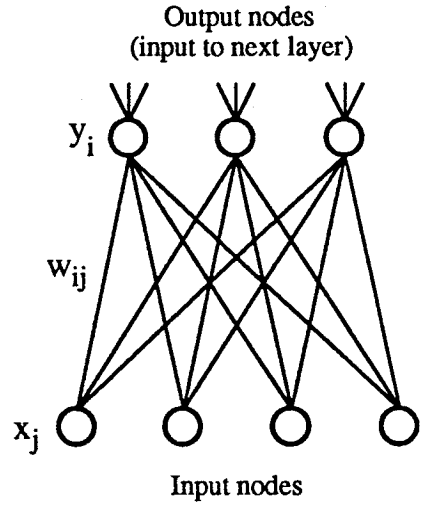

Figure 1: One layer of a first-order neural network.

The output of nodes in a general higher order network is given by:

$$y_i = \Theta(\Sigma_j w_{ij} x_j + \Sigma_j \Sigma_k w_{ijk} x_j x_k + \Sigma_j \Sigma_k \Sigma_l w_{ijkl} x_j x_k x_l + \ldots) \quad (2)$$

A diagram of a neural network utilizing only second-order terms is shown in Figure 2. Higher-order neural networks (HONNs) were evaluated in the 1960s for performing nonlinear discrimination but were rejected as impractical due to the combinatoric explosion of higher-order terms.[2]

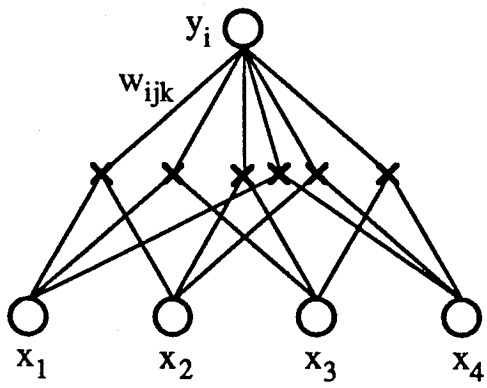

Figure 2: A second-order neural network with 4 inputs and 1 output.

At the same time, progress with multi-layer, first-order networks was restricted by the lack of an adequate learning rule. The advent of new learning rules,[3,4] of which the most popular is backward error propagation,[4] has allowed a great increase in the applications of multi-layer, first-order networks. However, their performance in terms of learning speed, the time taken to learn a training set of images, and generalization accuracy, the subsequent recognition of a separate test set, still prohibits their application to many practical problems.

In pattern recognition applications, one often desires the same response for an input image regardless of its position, size, and angular orientation. Achieving translation, scale and rotation invariance requires a neural network to learn relationships between the input pixels, x_j. Note that the summation within the parenthesis in Eq. (1) is a function of individual x_j's. No advantage is taken of any known relationships between the x_j's. Multi-layer, first-order networks can learn invariances, but require a great deal of training, and produce solutions that are specific to particular patterns. As an example, learning by back-propagation to distinguish a "T" from a "C", invariant to position and rotation, requires over 5000 presentations of an exhaustive training set.[5]

Building invariance into a HONN

Higher-order neural networks can also perform the nonlinear discrimination required for pattern recognition invariant to scale, translation, and rotation, but with considerable advantages over multi-layer, first-order networks. First, a HONN can perform nonlinear discrimination using only a single layer so that a simple perceptron learning rule can be used, leading to rapid convergence.[1] Further, the problem of combinatoric explosion can be overcome by building invariances into the network architecture using information about the relationships expected between the input x_j's.[6,7] The invariances achieved require no learning to produce and apply to any input pattern learned by the network.

As an example, translation invariance can be built into the second-order neural network with 4 input nodes and 1 output node shown in Figure 2. Assume that the input patterns (1 0 1 0) and (0 1 0 1) are to be indentified as the same object. If $w_{i13} = w_{i24}$, then y_i is the same for both inputs. In general, translation invariance requires that:

$$w_{ijk} = w_{i(j-k)} \quad (3)$$

i.e., the connections for equally spaced input pairs are all set equal.

Combinations of invariances can similarly be achieved. A second-order neural network will be simultaneously invariant to scale and

translation if the weights are set according to the function[7]

$$w(i,j,k) = w(i,(y_k - y_j)/(x_k - x_j)) \qquad (4)$$

Equation (4) implies that w_{ijk} is set equal to $w_{ij'k'}$ if the slope of a line drawn between nodes j and k equals that formed between j' and k', as shown in Figure 3. Any object drawn in a 2-D plane can have lines of various slopes drawn within it. An object's relative content of lines of different slopes does not change when it is translated in position or scaled in size, as long as it is not rotated.

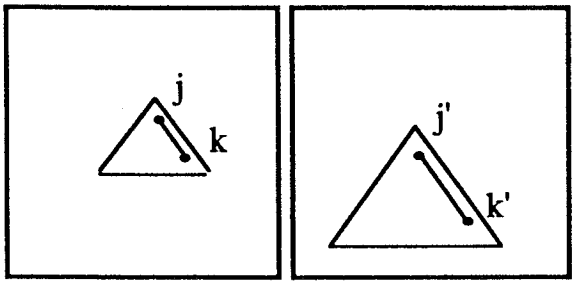

Figure 3: Translation and scale invariance achieved by setting $w_{ijk} = w_{ij'k'}$ if the slope of the line formed by nodes j and k equals that formed by nodes j' and k'.

Rotational invariance can be included by using a third-order neural network, where the output is given by the function

$$y_i = \Theta(\Sigma_j \Sigma_k \Sigma_l \; w_{ijkl} \; x_j \; x_k \; x_l) \qquad (5)$$

As shown in Figure 4, any three points within an object define a triangle with included angles (α,β,γ). When the object is translated, scaled and rotated, the three points in the same relative positions on the object still form the included angles (α,β,γ). Therefore, invariances to all three distortions can be achieved with a third-order network having an interconnection function of the form:

$$w_{ijkl} = w_{i\alpha\beta\gamma} = w_{i\gamma\alpha\beta} = w_{i\beta\gamma\alpha} \qquad (6)$$

Note that the order of angles matters, but not which angle is measured first.

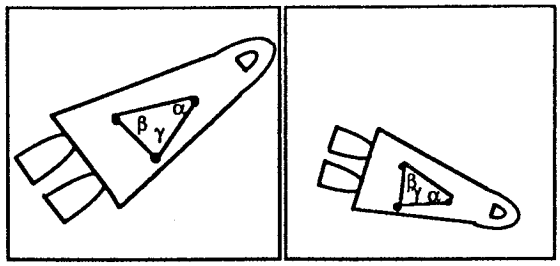

Figure 4: Translation, scale and rotation invariance is achieved by setting all third order weights equal for sets of inputs j, k, and l which form similar triangles.

Simulation results

We have simulated a single layer, second-order neural network using a 16 x 16, or 256 node, input field fully interconnected to a single output node which is thresholded with a hard limiter. There are 256-choose-2 or 32,640 input pairs and therefore interconnections. The interconnection weights are constrained to follow Eq. (4) in order to achieve invariance to scale and translation. The weights are initially set to zero and a perceptron learning rule is used:

$$\Delta w_{ijk} = (t_i - y_i) \, x_j \, x_k \qquad (7)$$

where the expected training output, t, actual output, y, and inputs x, are all binary. The network is trained on just 2 distinct patterns -- only one size and one location for each pattern. It learns to distinguish between the patterns in less than 1 minute of run time on a Sun 3 workstation. After training, it successfully distinguishes between all translated and scaled versions of the two objects with 100% accuracy. No further training is required to achieve this invariance, as it is built into the architecture. The system can learn to distinguish between any two distinct patterns, and has been tested on a variety of problems, including the T-C problem.[2] Scale invariance of a factor of 5 has

been demonstrated for this problem, with 100% recognition accuracy.

Due to the limited resolution of the finite 16 x 16 input window, residual scale variance can occur. (T,C) pairs are distinguished by their relative content of horizontal and vertical information. For the smallest (T,C) pair, shown in Figure 5a, the T has 3 input pair combinations arranged horizontally and 3 vertically, while the C has 2 arranged horizontally and 4 vertically. In the next larger scale of (T,C), shown in Figure 5b, the ratio of horizontal to vertical pixel pairs is 34:34 for the T and 26:42 for the C. It is therefore easier to distinguish between the smaller (T,C) pair based on their relative horizontal/vertical content. If the system is trained on the smaller set of letters, learning is not pushed to the point where larger versions can be recognized. In contrast, if large patterns are used for training, all smaller versions are subsequently recognized.

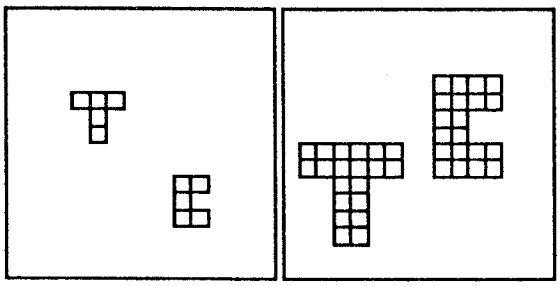

Figure 5: Two different scales of T and C drawn in a 16 x 16 pixel window.

Residual scale variance can be eliminated by using bipolar training values and a modified threshold function such as,

$$\Theta(x) = 1, \text{ if } x > K,$$
$$\Theta(x) = -1, \text{ if } x < -K, \qquad (8)$$
$$\Theta(x) = 0, \text{ otherwise,}$$

where K is some positive constant. Learning with a sufficiently large value for K forces the network to make a greater distinction between the initial patterns, allowing easier discrimination between test patterns which are subsequently evaluated with a hard limiter. Training the network on the smallest (T,C) pair using a value of K = 1000 allows correct identification of all larger test versions, without greatly increasing the training time.

Comparison to back-propagation

The second-order neural network was compared to a fully-connected, two layer, first-order network containing 256 input nodes, 128 hidden nodes, 1 output node, and therefore a comparable number (32,768) of interconnections. The first-order network was trained to perform scale and translation invariant recognition of two objects, the (T,C) pair, using a generalized back-propagation delta rule[4] with learning rate $\eta = 0.25$ and momentum factor $\alpha = 0.9$. After being shown 100 randomly selected pairs (out of 412 possible pairs which can be drawn in a 16x16 window) several thousand times, involving several days of run time on the Sun 3, the network failed to recognize more than half of the training set correctly. It is clear that this is only a preliminary comparison - a more sophisticated back-propagation model including a version of receptive fields[5] might be expected to perform scale and translation invariant recognition, but such a model is unlikely to compare favorably with the speed and accuracy demonstrated by the second-order neural network described above.

Conclusions

Higher-order neural networks can be designed to perform pattern recognition, invariant to translation, scale and rotation distortions. The large number of connections in a HONN are used to advantage by redundantly encoding invariant information into the network architecture. The speed of learning with a single layer perceptron more than compensates for the large number of weights to be learned. Further, as the network is only trained on one example of each object to be recognized, most of the weights in the network are not modified directly during

learning. Instead, sets of weights are constrained to be equal, so that changing one weight effectively changes many weights at once.

Our simulations have demonstrated that a second-order neural network can be rapidly trained to distinguish between two patterns regardless of their size and translational position. 100% recognition accuracy was achieved for several different training pattern pairs using a 16 x 16 input field size. Preliminary comparisons show the HONN to be vastly superior to a two layer, first-order network trained by back-propagation in terms of learning speed and recognition accuracy. The next step in testing the capability of the HONN model will be to achieve simultaneous invariance to rotation, scale and translation by using a third order network.

References

[1] F. Rosenblatt, Principles of Neurodynamics. New York: Spartan, 1962.

[2] M.L. Minsky and S. Papert, Perceptrons. Cambridge, MA: MIT Press, 1969.

[3] P. Werbos, Beyond Regression: New T. Ph.D. thesis, Harvard University, 1974 (unpublished).

[4] D.E. Rumelhart, G.E. Hinton, and R.J.Williams, "Learning Internal Representations by Error Propagation," in Parallel Distributed Processing, (D.E. Rumelhart and J.L.McClelland, eds.), vol. 1, ch. 8, Cambridge, MA: MIT Press, 1986.

[5] *Ibid.*, pp. 348-352.

[6] C.L. Giles and T. Maxwell, "Learning, invariance, and generalization in high-order neural networks," Applied Optics, vol. 26, pp. 2972-4978, 1987.

[7] C.L. Giles, R.D. Griffin, and T. Maxwell, "Encoding geometric invariances in higher-order neural networks," in Neural Information Processing Systems, American Institute of Physics Conference Proceedings, D.Z. Anderson, ed., p. 301, 1988.